Attorney's Docket No.: 10559/614001 / P12853

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: QUEUE MANAGEMENT

APPLICANT: GILBERT WOLRICH, MARK B. ROSENBLUTH, DEBRA BERNSTEIN AND DONALD F. HOOPER

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL870691145US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the U.S. Patent and Trademark Office, Washington D.C. 20231.

December 12, 2001
Date of Deposit

Signature

Gabe Lewis
Typed or Printed Name of Person Signing Certificate

QUEUE MANAGEMENT

TECHNICAL FIELD

This invention relates to managing a queue structure and more specifically to scheduling the transmission of packets on an electronic network.

BACKGROUND

Electronic networks perform data transfers using a variety of data packet sizes. A packet size may be larger than the input or output capacity of a device connected to the network. Therefore a single packet may require multiple transfers of smaller "cells" to completely transfer the packet.

DESCRIPTION OF THE DRAWINGS

FIG.1 is a block diagram of computer hardware on which a queue management process may be implemented.

FIG. 2A is a block diagram representing an exemplary linked queue array.

FIG. 2B is a block diagram representing addresses stored in a linked queue array being mapped to stored data packets.

FIG. 2C is a flowchart representing the transmission of data packets.

FIG. 3 is a flowchart showing a queue management process.

FIG. 3A is a flowchart showing an en-queuing process.

FIG. 3B is a flowchart showing a de-queuing process.

DESCRIPTION

Referring to FIG. 1, a network processing system 10 is shown operating as a data packet cross-bar device. The network processing system 10 receives packets (through I/O buses 14a-14n from external devices not shown), stores the packets temporarily, interprets header (address) information contained in each packet, and transmits each packet to a destination indicated by its header when an appropriate I/O bus 14a-14n is available. System 10 may include connections to thousands of I/O buses and may need to simultaneously store and track tens of thousands of data packets of various sizes before each packet is transmitted out of system 10. The storage, and the input and output of data packets (packets) to and from I/O buses 14a-14n is controlled by several processors 12a-12n.

System 10 includes a first memory 18, to store the received data packets from a set of data buffers 18a-18n. The data buffers 18a-18n are not necessarily contiguously stored in the first memory 18. Each data buffer is indexed by a

buffer descriptor address (BDA) that indicates the location and size of the buffer. As each packet is received from one of the I/O buses and stored by one of the processors 12a-12d in one of the buffers of the first memory 18, the processor, e.g., 12a identifies one of a set of I/O ports 16a-16n for transmitting the packet from the data buffer 18a-18n out of system 10. Each of the I/O ports is associated with one of the I/O buses).

Often, the I/O port chosen for transmitting a packet stored in a buffer is busy receiving or sending packets for other buffers. In this case, the system 10 includes a second memory 20 for storing the packet. The second memory 20 stores a queue array 24. The queue array 24 has buffer descriptor addresses (BDAs) for packets that are stored in buffers 18a-18n of the first memory 18 and are waiting for an assigned I/O port 16a-16n to become available.

Each data packet received may vary in size. Therefore, the size of each data buffer 18a-18n may also vary. Furthermore, each data buffer 18a-18n may be logically partitioned by a processor 12a-12n into one or more "cells". Each cell partition represents a maximum size of a data packet that may be transmitted by an I/O buffer 16a-16n. For example, in FIG. 1, buffer 18a is partitioned into two cells, buffer

18b includes only one cell, and buffer 18c includes three cells.

System 10 also includes queue manager logic 22 connected to processors 12a-12n and second memory 20. Queue manager 22 includes a queue array 24 that includes several queue entries, with each queue entry corresponding to an I/O buffer, 16a-16n. Each queue entry in queue array 24 stores multiple BDAs, where one BDA links to another BDA in the same queue. Queue array 24 is stored in second memory 20. Alternatively or in addition thereto the queue manager 22 may include a cache containing a sub-set of the contents of queue array 24.

Each BDA includes both an address of the stored data buffer in first memory 18, and a "cell count" that indicates the number of cells contained in a data buffer, 18a-18n. The BDA is, for example, 32 bits long, with the lower 24 bits being used for address of the buffer descriptor and the upper 7 bits being used to indicate the cell count of the data buffer.

Processors 12a-12n store and retrieve data buffers from queue array 24 by sending "En-queue" and "De-queue" commands to queue manager 22. Each En-queue and De-queue command includes a queue entry number included in queue array 24. Queue manager 22 responds to a De-queue command by returning a

BDA stored at the "head" , i.e., the top entry of the queue entry specified to the requesting processor. Queue manager 22 also uses the cell count included in the head BDA being returned to determine whether all of the cells included in the corresponding data packet have been sent. If the cell count is greater than zero, then the queue manager leaves the head BDA in the head location of the queue. When the cell count for a De-queued BDA has reached zero another linked BDA is moved to the head of the queue, as will be explained.

Referring to FIGS. 2A and 2B, a first queue entry, "Qa" is shown of an exemplary queue array Qa-Qn is shown. Each queue entry included in queue array Qa-Qn includes a three-block queue descriptor 51a-51n, and may also include additional BDAs that are linked to the same queue entry. Each queue descriptor 51a-51n includes a first block 52a-52n that contains the head BDA for the queue entry, a second block 54a-54n that contains the "tail" address for the queue entry and a third block 56a-56n that contains a "queue count" for the queue entry.

As an example of a queue entry that includes both a head BDA and a linked BDA, queue "Qa" is shown in FIG. 2A. In this example, head block 52a has the BDA that will be returned in response to a first De-queue command specifying entry Qa. Head

BDA 52a links to a second BDA stored at address "a:" 57a. "Tail" block 54a contains the address "b:" of the last linked address of entry Qa. The address contained in Tail block 54a points to the storage location where another BDA may be En-Queued (and linked to) queue entry Qa. Third block 56a contains the current value of Queue Count that indicates the number of linked buffer descriptors included in the Q. In this example, Queue Count equals two, indicating a first BDA in the "head" location 52a and a second linked BDA in block 57a. It is noted that the BDA contained in the head block 52a-52n, of each queue descriptor 51a-51n, contains the BDA and Cell Count of the second linked BDA on the Q, 57a-57n, unless the Q includes only a single BDA.

Referring to FIG. 3, a process 80 is shown for En-queueing BDAs and linking the BDAs to subsequent BDAs using the queue array shown in FIGS. 2A and 2B. Process 80 includes a sub-process 100 that depicts En-queueing a BDA onto a queue array structure, and a sub-process 120 that depicts De-queueing a BDA from a queue array.

Referring to FIG. 3A, a example of the sub-process 100 receives (102) an En-queue command that specifies a Q in the queue array Qa-Qn and a BDA for a new data buffer. Sub-process stores (104) the new BDA in the location indicated by the tail

address, up-dates (106) the tail address to point to the new BDA and increments (108) the queue count by one (block 56a-56n). Sub-process 100 may be repeated to store and link, additional data buffers onto the "tail" of a queue entry, that is, En-queueing an additional BDA onto the linked address location at the tail of a queue entry, etc.

Referring to FIG. 3B, an example of sub-process 120 depicts a process of De-queueing data buffers, i.e., BDAs, from a queue entry included in queue array Qa-Qn (see FIG. 2B). Sub-process 120 receives (122) a De-queue command that specifies a queue entry included in queue array Qa-Qn. Sub-process 120 returns (122) the BDA from the head of the queue descriptor for the specified queue entry to the requesting processor. Process 120 determines (126) whether the cell count from the head BDA is greater than zero. If the cell count is greater than zero, process 120 decrements (128) the cell count included in the head BDA and exits (140). If the cell count is not greater than zero, process 120 determines (129) if the Queue Count is greater than or equal to one. If the Queue Count is not greater than or equal to one (indicating the queue entry is empty) process 120 exits (140). If the Queue Count is determined (129) greater than or equal to one (indicating the queue entry contains another linked BDA)

process 120 sets (130) the next linked BDA to be the head buffer descriptor and exits (140). Sub-process 120 may be repeated to De-queue the head BDA and subsequent linked BDAs stored in a queue entry in queue array 24.

Performing process 80 on a system, such as system 10, enables the system to keep a multiple-cell data buffer that is being transmitted at the head of a queue entry. This is an advantage when a relatively large number of I/O ports are being served concurrently, with one or more I/O ports requiring cell-at-a-time data transmission.

Referring to FIG. 4, a logical representation of the sequence of data packets that are transmitted by a system performing process 80 is shown. The data buffers, 18a-18n, and Cell numbers of FIG. 4 correspond to the same numbers shown in Figs. 1 and 2B. As shown in FIG. 4, a system performing process 80 causes the two cells of data buffer 18a to be transmitted before the transmission of the first Cell of data buffer 18b is begun. Likewise, data buffer 18b completes transmission before the first cell of data buffer 18c begins transmission, and so forth.

FIG. 1 shows a computer system 10 on which process 80 may be implemented. Process 80 is not limited to use with the hardware and software of FIG. 1. It may find applicability in

any computing or processing environment. Process 80 may be implemented in hardware, software, or a combination of the two. Process 80 may be implemented in computer programs executing on programmable computers or other machines that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage components), at least one input device, and one or more output devices. Program code may be applied to data entered using an input device (e.g., a mouse or keyboard) to perform process 80 and to generate output information.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on a storage medium/ article (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform process 80. Process 80 may also be implemented as a machine-readable storage medium, configured with a computer program,

where, upon execution, instructions in the computer program cause a machine to operate in accordance with process 80.

The invention is not limited to the specific embodiments described above. For example, a single memory may be used to store both data packets and buffer descriptors. Also, the buffer descriptors and BDAs may be stored substantially simultaneously in second memory 20 and queue array 24 (see FIG. 1).

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is: